Northeastern University
College of Engineering

EECE 5698 - ST: Reinforcement Learning Instructor: Prof. Mahdi Imani

Project #3 (Due Date: April 30)

Problem 1:

Consider Problem 1 described in Project 2. The goal is to analyze the performance of the temporal difference learning algorithms.

Set $p = 0.02$, $\gamma = 0.95$, $\alpha = 0.3$, $\epsilon = 0.1$, the total number of episodes 1,000, the maximum number of steps if the agent does not reach the goal state 1,000 (i.e., the maximum length of each episode). Any parameter that is not specified can be set/tuned by you, and you need to include them in your final report.

Implement Q-Learning, SARSA and actor-critic algorithm ($\beta = 0.05$) algorithms. For each algorithm, you need to run the algorithm for 10 independent runs. Report your results in terms of:

a) How many times among 10 independent runs, upon the termination of learning, a path from start to goal has been obtained?
b) Show the optimal policy and the optimal path from start to goal for one of the 10 independent runs (use the results of a run that the path from start to goal is found; if the path is not found in all runs, show the results from a random run). See Figures 1 and 2.
c) Show the average accumulated reward (in 10 independent runs) with respect to the episode number. See Figure 3.
d) Plot the average accumulated reward with respect to the episode number for all algorithms in a single plot. Describe your findings.

*For any of the algorithms that in all runs the path from start to goal is not found, you need to change the parameters (e.g., α, β, ϵ, episode number, length of the episode) in a trial and error and report the best solutions found.

SARSA Algorithm

Initialize $Q(s, a)$ arbitrarily
Repeat (for each episode):
. Initialize s
. Choose a from s using policy derived from Q
. . . (e.g., ϵ -greedy)
. Repeat (for each step of episode):
. . Take action a, observe r, s'
. . Choose a' from s' using policy derived from Q
. . . (e.g., ϵ -greedy)
. . $Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma Q(s', a') - Q(s, a)]$
. . $s \leftarrow s'$; $a \leftarrow a'$;
. until s is terminal

Q-Learning Algorithm

Initialize $Q(s, a)$ arbitrarily
Repeat (for each episode):
. Initialize s
. Repeat (for each step of episode):
. . Choose a from s using policy derived from Q
. . . (e.g., ϵ -greedy)
. . Take action a, observe r, s'
. . $Q(s, a) \leftarrow Q(s, a) + \alpha\left[r + \gamma \max_{a'} Q(s', a') - Q(s, a)\right]$
. . $s \leftarrow s'$;
. until s is terminal

Tabular Actor-Critic Algorithm

$V(s)=0$, $H(s,a)=0$, for all $s \in \mathcal{S}$, $a \in \mathcal{A}$.

Repeat for N episodes

- Start from a random state $s_0 \in \mathcal{S}$, $t=0$

  While $t<T$ (episode length):

  - Select action: $a_t \sim \pi(\cdot|s_t)$: $\pi(a|s)=\frac{e^{H(s,a)}}{\sum_{a' \in A} e^{H(s,a')}}$
  - Take action a_t, move to state s_{t+1} and observe R_{t+1}.
  - $\delta_t = R_{t+1} + \gamma V(s_{t+1}) - V(s_t)$
  - $V(s_t) = V(s_t) + \alpha \delta_t$
  - $H(s_t, a_t) = H(s_t, a_t) + \beta \delta_t (1 - \pi(a_t|s_t))$
  - $t = t+1$

SARSA(λ) Algorithm

Initialize $Q(s,a)$ arbitrarily and $e(s,a)=0$, for all s,a
Repeat (for each episode):
  Initialize s, a
  e(s,a)=0, for all s,a
  Repeat (for each step of episode):
    Take action a, observe r, s'
    Choose a' from s' using policy derived from Q (e.g., ε-greedy)
    $\delta \leftarrow r + \gamma Q(s',a') - Q(s,a)$
    $e(s,a) \leftarrow e(s,a) + 1$
    For all s,a:
      $Q(s,a) \leftarrow Q(s,a) + \alpha \delta e(s,a)$
      $e(s,a) \leftarrow \gamma \lambda e(s,a)$
    $s \leftarrow s'$; $a \leftarrow a'$
  until s is terminal

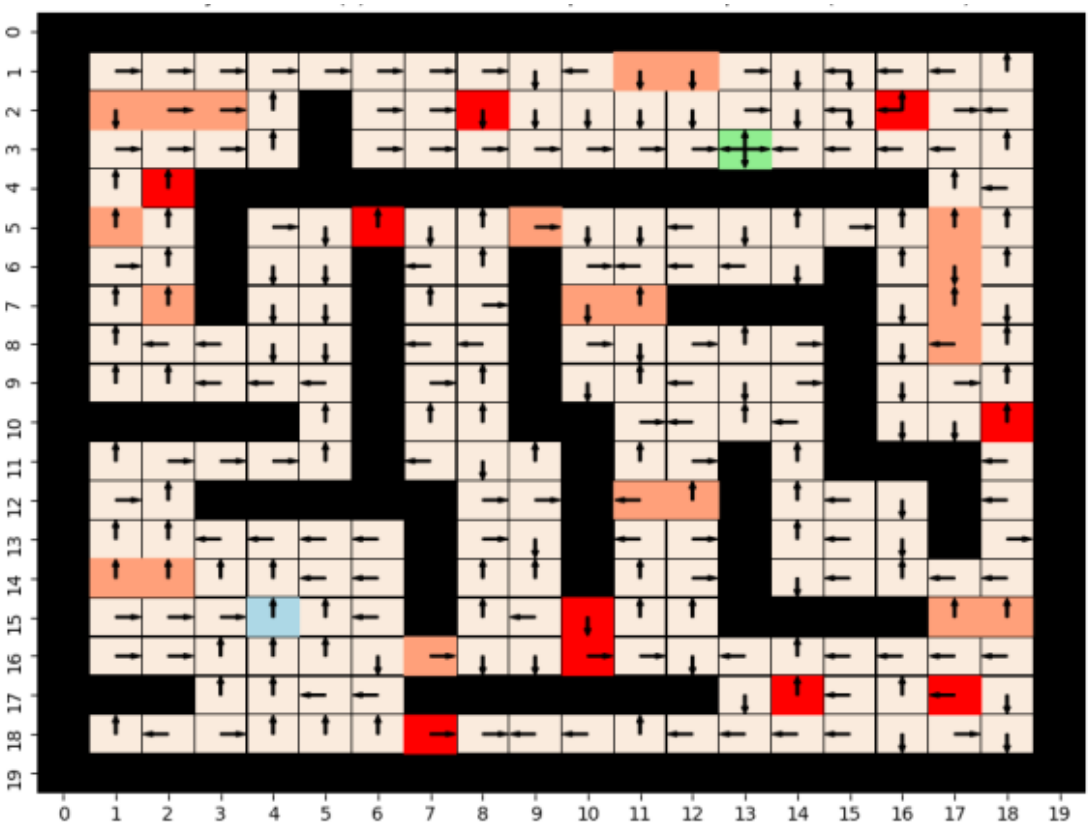

Figure 1 The optimal policy obtained by SARSA for alpha=0.3 and epsilon=0.1.

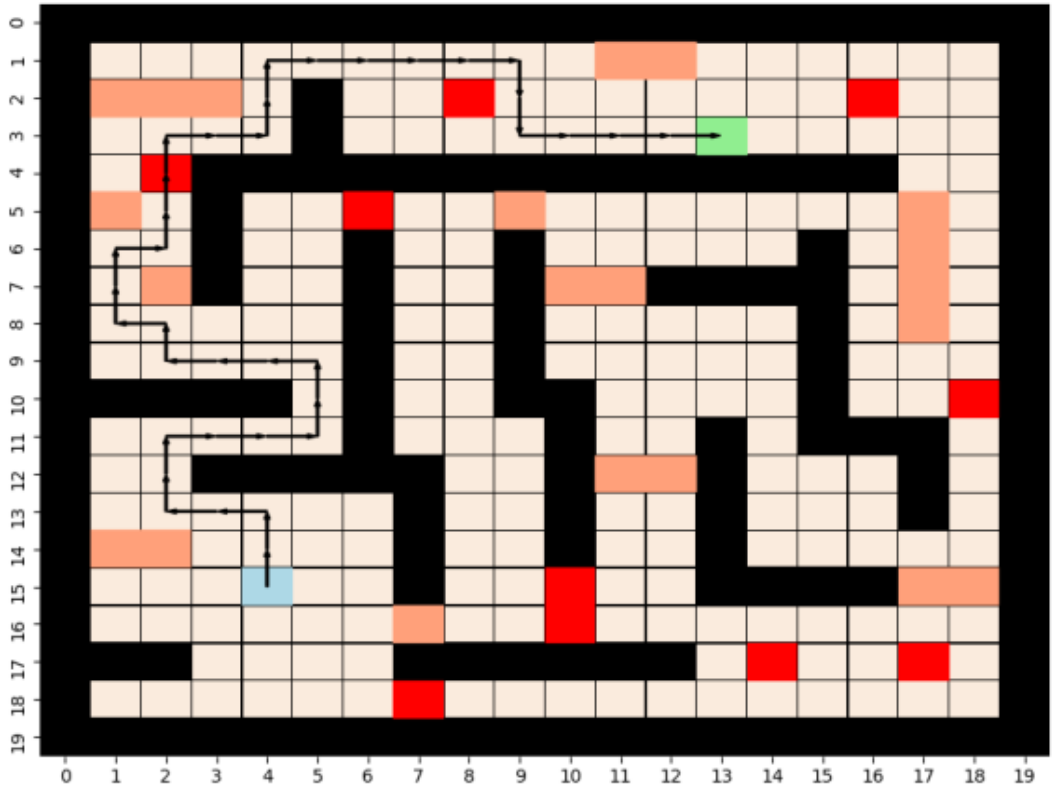

Figure 2 The optimal path obtained by SARSA for alpha=0.3 and epsilon=0.1.

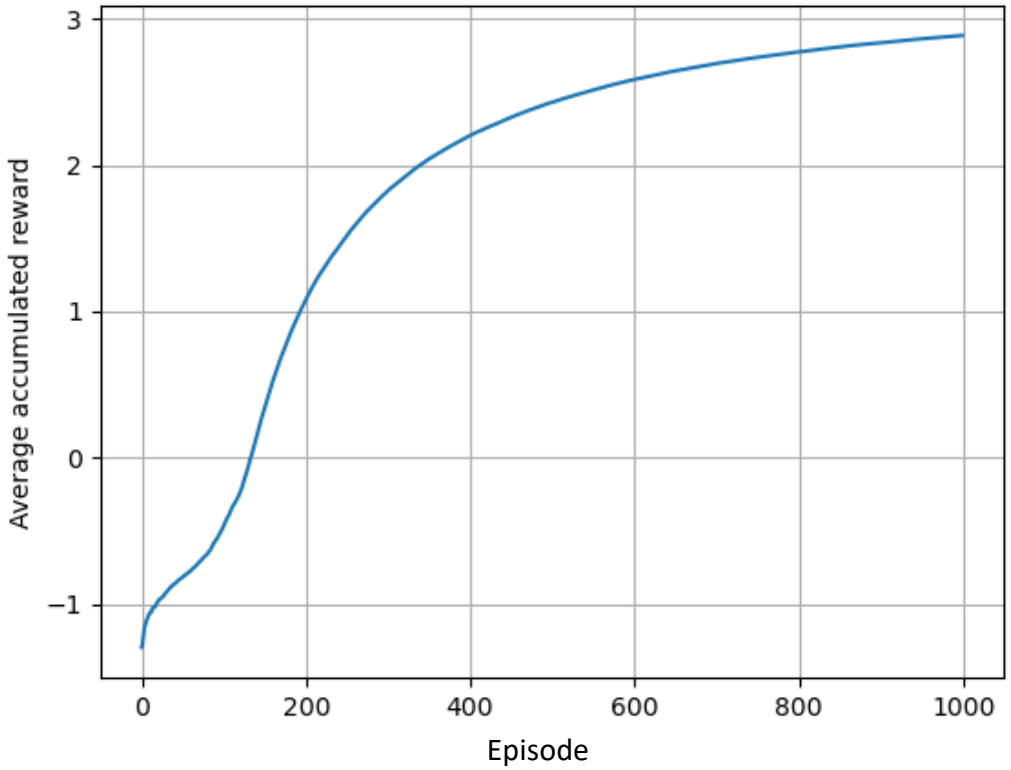

Figure 3 The average accumulated reward with respect to the episode obtained by SARSA for alpha=0.3 and epsilon=0.1.

Problem 2

Consider Problem 2 described in Project 2. The state space consists of $2^4 = 16$ states and the action space is $A = \{\boldsymbol{a}^1 = [0,0,0,0]^T, \boldsymbol{a}^2 = [0,1,0,0]^T, \boldsymbol{a}^3 = [0,0,1,0]^T, \boldsymbol{a}^4 = [0,0,0,1]^T\}$. The dynamics of this system is governed by the following equation:

$$\boldsymbol{s}_k = \overline{\underbrace{\begin{bmatrix} 0 & 0 & -1 & 0 \\ +1 & 0 & -1 & -1 \\ 0 & +1 & 0 & 0 \\ -1 & +1 & +1 & 0 \end{bmatrix}}_{\text{C: Connectivity Matrix}} \boldsymbol{s}_{k-1} \oplus \boldsymbol{a}_{k-1}}} \oplus \boldsymbol{n}_k \quad \text{Eq. (1)}$$

where $\overline{\boldsymbol{v}}$ maps the element of vector greater than 0 to 1, and smaller or equal to zero to 0, $\boldsymbol{a}_{k-1}$ is a control input and $\boldsymbol{n}_k$ is the state transition noise.

We assume that four elements of $\boldsymbol{n}_k$ are identically distributed through a Bernoulli noise with parameter p. This can be expressed as: $\boldsymbol{n}_k(j) \sim \begin{cases} 1 & with\ prob\ p \\ 0 & with\ prob\ 1-p \end{cases}$, for $j = 1, \dots, 4$.

The reward function is as follows:

$$R(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{s}') = 5\ \boldsymbol{s}'^{(1)} + 5\ \boldsymbol{s}'^{(2)} + 5\ \boldsymbol{s}'^{(3)} + 5\boldsymbol{s}'^{(4)} - |\boldsymbol{a}|$$

where $|\boldsymbol{a}|$ sums the absolute value of the elements of the vector $\boldsymbol{a}$. According to this reward function, each activation of a gene has the reward 5 and actions $\boldsymbol{a}^2$ and $\boldsymbol{a}^3$ have reward -1.

For creating state-action-reward transitions needed in temporal difference learning algorithms, you need to use Eq. (1). Let's consider the current state is $\boldsymbol{s}_k = [0\ 1\ 1\ 1]^T$, if the action $\boldsymbol{a}_k = [0\ 0\ 0\ 1]^T$ is selected, you need to create $\boldsymbol{n}_k$ by drawing 4 indepdent samples from Bernoulli (p).

For simplicity, let's assume $\boldsymbol{n}_k$=[0,0,0,0]^T. Then the next state can be computed according to Eq. (1) as:

$$\boldsymbol{s}_{k+1} = \overline{\begin{bmatrix} 0 & 0 & -1 & 0 \\ +1 & 0 & -1 & -1 \\ 0 & +1 & 0 & 0 \\ -1 & +1 & +1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 1 \\ 1 \end{bmatrix}} \oplus \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \oplus \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \overline{\begin{bmatrix} -1 \\ -2 \\ +1 \\ +2 \end{bmatrix}} \oplus \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \oplus \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \end{bmatrix} \oplus \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \oplus \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

Set $p = 0.05, \gamma = 0.95$, α=0.2, $\epsilon = 0.1$, the total number of episodes 500, the maximum number of steps (i.e., the maximum length of each episode) 500.

Implement Q-Learning, SARSA , SARSA($\lambda = 0.95$) and actor-critic ($\beta = 0.05$) algorithms. For each algorithm, you need to run the algorithm for 10 independent runs.

a) Show the optimal policy for all 10 independent runs (10 vectors of size 16, consisting of $\boldsymbol{a}^1$,$\boldsymbol{a}^2$ or $\boldsymbol{a}^3$)
b) Show the average accumulated reward (in 10 independent runs) with respect to the episode number.
c) Plot the average accumulated reward with respect to the episode number for all algorithms in a single plot. Describe your findings.

* For any of the algorithms that in all runs the path from start to goal is not found, you need to change the parameters (e.g., $\alpha, \beta, \lambda, \epsilon$, episode number, length of the episode) in a trial and error and report the best solutions found.

Questions about the project should be directed to TA, Begum Taskazan, at taskazan.b@northeastern.edu.